Geometria Computacional

MAC 0331

Lista 5

2 Seja P um polígono y-monótono com n vértices. Suponha que temos em dois vetores $e[0..k-1]$ e $d[0..n-k+1]$ os índices dos vértices de P na cadeia esquerda e direita, ordenados por y-coordenadas.

Escreva um algoritmo que, dado n, P, k, e, d, decide se um dado ponto q pertence a P ou não. A complexidade de tempo do seu algoritmo deve ser $O(\lg n)$.

Vamos dividir o plano em várias faixas horizontais e fazer uma busca binária para saber em qual faixa o ponto está. Essas faixas serão dadas pelos pontos (como se fosse uma trapezoidação). Depois, testamos se o ponto está a esquerda da cadeia direita e a direita da cadeia esquerda

```
Pertence (n, P, k, e, d, q)

    aresta_esq ← Busca (k-1, e, q)

    aresta_dir ← Busca (n-k+1, d, q)

    se aresta_esq = NIL ou aresta_dir = NIL
        então devolva falso

    se esquerda(aresta_dir, q) e direita(aresta_esq, q)
        então devolva verdadeiro

    devolva falso
```

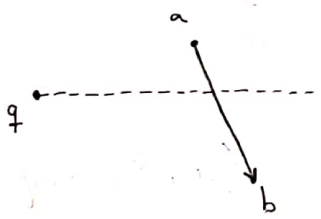
7 (a) Modifique o algoritmo de ray crossing para evitar a única operação com reais. Use uma das primitivas vistas

Nós fazemos a conta com os Reais para descobrir onde o raio cruza a aresta, para então verificar se a aresta está a direita do ponto.

Então podemos apenas trocar essa operação pela primitiva

ESQUERDA+(a, b, c), que indica se o ponto c está exclusivamente a esquerda da reta dada por $\overrightarrow{ab}$, simétrico para DIREITA+(a,b,c).

Só temos que garantir que o vértice a é sempre o que está acima




```
Em_Poligono (P, n)          ▷ considerando q = (0,0)
1    c ← 0        d ← 0
2    para i ← 0 até n-1 faça
3        se P[i][X] = 0 e P[i][Y] = 0 então devolva vértice
4        j ← (i+n-1) mod n
5        teste_c ← (P[i][Y] > 0) ≠ (P[j][Y] > 0)
6        teste_d ← (P[i][Y] < 0) ≠ (P[j][Y] < 0)
★ 7      se P[i][Y] > P[j][Y]
★ 8          então a ← P[i]    b ← P[j]
★ 9          senão a ← P[j]    b ← P[i]
★ 10     se teste_c ou teste_d
```

```
* 11          então se testeE ESQUERDA+(a,b,c)
* 12                  então c ← c+1
* 13              se testeD DIREITA+(a,b,c)
* 14                  então d ← d+1
  15      se c e d tem paridade distinta
  16          então devolva em aresta
  17      se c é ímpar
  18          devolva dentro
  19      devolva fora
```

(b) Use uma das primitivas para detectar se o ponto está numa das arestas, evitando assim a necessidade da segunda versão do algoritmo.

```
Em-Poligono(P, n)
1   c ← 0
2   para i ← 0 até n-1 faça
3       se P[i][X] = 0 e P[i][Y] = 0
4           então devolva vértice
5       j ← (i+n-1) mod n
6       se ENTRE(P[i], P[j], (0,0))
7           então devolva em aresta
8       se (P[i][Y] > 0 e P[j][Y] ≤ 0) ou (P[i][Y] ≤ 0 e P[j][Y] > 0)
9           então se P[i][Y] > 0
10                  entao a ← P[i]   b ← P[j]
11                  senão a ← P[j]   b ← P[i]
12              se ESQUERDA(a,b,c)
13                  então c ← c+1
```

```
14  se c é impar
15  |     entao devolva dentro
16  devolva fora
```